View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Calhoun, Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Faculty and Researcher Publications

Faculty and Researcher Publications

1997

Model Management in Electronic Markets for Decision Technologies: A Software Agent Approach

Bhargava, Hemant K.

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

Model Management in Electronic Markets for Decision Technologies: A Software Agent Approach *

Hemant K. Bhargava
Code SM-BH
Naval Postgraduate School
bhargava@nps.navy.mil

Ramayya Krishnan
The Heinz School
Carnegie Mellon University
rk2x@andrew.cmu.edu

Stephen Roehrig
The Heinz School
Carnegie Mellon University
roehrig+@cmu.edu

Michael Casey
Naval Postgraduate School
mpcasey@nps.navy.mil

David Kaplan
The Heinz School
Carnegie Mellon University
dk3g@cmu.edu

Rudolf Müller
Institut für Wirtschaftsinformatik
Humboldt-Universität Berlin
rmueller@wiwi.hu-berlin.de

Abstract

DecisionNet is a distributed, Web-based electronic market for decision technologies such as data, models, solution algorithms, and modeling environments. Consumer-provider interactions are facilitated by model management software agents provided by DecisionNet. To illustrate different approaches for designing this agent functionality, we present two agents that embody different designs for mediating consumer and provider interaction with the AMPL and GAMS modeling environments. The AMPL agent is lean, and places significant knowledge and reasoning requirements on both providers (when registering a technology) and consumers (when using technologies). In contrast,the GAMS agent encapsulates knowledge of the GAMS language and modeling environment to facilitate registration of models by providers and to create a run time interface to models for consumers. We discuss the relative advantages of both approaches and argue for the need to incorporate them into environments such as DecisionNet.

*The authors acknowledge funding by the National Science Foundation (grant IRI-9312143), the U.S. Army Artificial Intelligence Center (grant MIPR6GNGS00087), U.S. Army Research Office (grant DAAH04-94-G-0239), the Institute for Joint Warfare Analysis at the Naval Postgraduate School, and the National Research Center SFB 373 of the Deutsche Forschungsgemeinschaft.

1 Introduction

The explosive growth of the World Wide Web [3] creates new opportunities for the development and deployment of decision technologies for and by organizations and individuals [1, 2, 9, 10, 11]. The DecisionNet project [4, 5] aims to improve the usability, interoperability, and reusability of decision technologies, by exploiting this growth of the Web. DecisionNet is available on the World Wide Web at http://dnet.sm.nps.navy.mil/.

Fundamental to the DecisionNet concept is the idea that decision technologies be treated as objects (*services*) that can be *used* without having to be *owned* by consumers, thus adding a *use* option to the *make* and *buy* options that potential consumers traditionally have. While DecisionNet presently is a non-commercial research prototype, a logical conclusion of the DecisionNet idea is the establishment of an electronic marketplace for the dissemination and use of decision technologies.

To a user of decision technologies, DecisionNet is a distributed collection of decision technologies, each of which is accessible and executable over a global network, in this case the World Wide Web. The technologies in DecisionNet are owned and maintained by various technology providers on their own computational platforms, and are listed in the collection through a registration process directed by software agents. Tech-

1060-3425/97 $10.00 © 1997 IEEE

registration; execution occurs on the provider's machines, and problem-specific input and output data is exchanged via HTML (HyperText Markup Language) forms, e-mail, or the Internet's file transfer protocols. These agent features and capabilities may be thought of as model management procedures in a distributed modeling setting.

In this paper, we focus on the the analysis and design of these software agents. The requirements for agent functionality may be understood by examining the following questions.

What is the consumer's level of knowledge about decision technologies?

The level of support that DecisionNet agents are programmed to provide to consumers is a function of the assumptions made about the knowledge that the consumer has about decision technologies.

What is the technology available to the consumer?

The technology available to the consumer determines the interface to the technologies available in DecisionNet.

What are the technical capabilities of decision technology providers?

Since technologies are maintained by providers on their computational platforms, the degree of support desired or required by these providers to create interfaces to their technologies determines agent functionality.

What role is to be played by DecisionNet? Are these agents to be implemented by technology providers or are they to be implemented as an infrastructural service by DecisionNet?

Agents provided by DecisionNet as an infrastructural service can reduce barriers to entry into DecisionNet for both consumers and providers. They can also mediate transactions and provide a set of centralized services such as internet protocol assistance and billing. However, their development can constitute significant effort, and providers with the capability may choose to develop their own agents.

To provide substantive context, we discuss the analysis and design of two software agents that facilitate interaction with optimization models and associated modeling environments (AMPL [8] and GAMS [6, 7]) in DecisionNet. These agents embody different designs and are based on a different set of assumptions and answers to the questions listed above.

The rest of the paper is organized as follows. We begin with two simple examples in §2 which we use architecture and discuss two key transactions: the addition of decision technologies by providers and the execution of these technologies by consumers. We then discuss, in §4 and §5, the design and analysis of two software agents—the *AMPL agent* and the *GAMS agent*, respectively. In each case, in addition to the technical issues, we also discuss the implications their designs have for barriers to entry into DecisionNet for consumers and providers. We conclude with thoughts on future research issues.

2 Problem Scenarios

2.1 Example I

A small-sized University is faced with a final examination scheduling problem. An analyst at the registrar's office responsible for the creation of the schedule decides to develop a mathematical programming model to solve her problem. She formulates the model and states it using the AMPL modeling language [8]. She develops a data file required to instantiate the model in the AMPL syntax. She now looks for an AMPL environment to solve her model in the DecisionNet yellow pages on the World Wide Web. Upon searching the yellow pages, she discovers an **AMPL modeling environment**. *She browses through some descriptive information about it and is satisfied that it can be used to solve her model. She double clicks on the entry in the DecisionNet yellow pages. She receives an HTML form which requests the location of the AMPL model file, data file and command file. She moves the files she has developed to her FTP (File Transfer Protocol) server and supplies the URL of these files in the form. She submits this request and after a short period of time receives the result of the AMPL execution. She examines the results, makes some modifications to the AMPL model and data files and resubmits the files for execution. Satisfied with the results, she saves the results and creates an examination schedule using the results of her model.*

Let us summarize the main features of this scenario in terms of the questions presented in §1. We will use this characterization to motivate the design of the AMPL agent in DecisionNet.

What is the consumer's level of knowledge about decision technologies?

In this scenario, the consumer is knowledgeable about mathematical programming methodology, models, and modeling languages such as AMPL. The analyst formulates the model, states it in the AMPL lan-

ing environment.

What is the technology available to the consumer?

In addition to a Web browser, the analyst has an anonymous FTP server available to her. She places the AMPL files that she has developed on this server in order to make it available for execution on the AMPL environment listed in the DecisionNet pages.

What are the technical capabilities of decision technology providers?

The scenario does not discuss this issue in detail. However, it is clear that the provider, in this case of the AMPL modeling environment, has developed an HTML form-based interface to the environment. Further, the provider has developed an internet assistant that can fetch files from remote FTP servers and invoke the AMPL environment with the appropriate parameters.

What role is played by DecisionNet?

The role played by DecisionNet is to advertise the availability of the AMPL modeling environment and to provide access to it.

2.2 Example II

A small-sized University is faced with a final examination scheduling problem. An analyst at the registrar's office responsible for the creation of the schedule finds the DecisionNet yellow pages on the World Wide Web. Upon searching the yellow pages, she discovers a **examination scheduling model.** *She browses through some descriptive information about the model and is satisfied that the model can be used to develop the examination schedule. She then requests execution of the model. A software agent leads her through a session in which she supplies requested data through a series of HTML forms. Following that the model is executed and the results of the execution returned to her Web browser. She examines the result file, makes some changes to the data and executes the model again. She saves the result file and uses it to create an examination schedule using the spreadsheet on her desktop.*

Once again, let us summarize the main features of this scenario using the set of questions presented in §1. Again, we use this characterization to motivate the design of the GAMS agent in DecisionNet.

What is the consumer's level of knowledge about decision technologies?

In this scenario, the consumer *searches for an application* to solve her problem. In contrast to the first cisionNet pages. Given her lack of knowledge about the modeling language in which the model is stated (say, GAMS), data required to instantiate the model is supplied by the analyst using HTML forms. In fact, the analyst may not even know the modeling language in which the examination scheduling model is stated or that it requires a particular kind of modeling environment to execute it.

What is the technology available to the consumer?

The only technology available to the analyst is a Web browser. The data required to instantiate the model is supplied using HTML forms.

What are the technical capabilities of decision technology providers?

The scenario does not discuss this issue in detail. In contrast to the first scenario, there are at least two decision technologies needed by the analyst – the model that she chose and the modeling environment required to execute it. These are maintained by the providers on their own computational platforms. The scenario indicates that the analyst interacts with the technologies in a transparent manner.

What role is to be played by DecisionNet?

Beyond advertising, DecisionNet implements the agent with which the analyst interacts. This is the agent that elicits the data and enables the transparent execution of the chosen model on the modeling environment.

2.3 Discussion

In each of these scenarios, model management support is provided. In the first case, the support is in the form of fetching each of the files from remote locations and assembling them in order that the modeling environment may be invoked correctly. These tasks are performed by the AMPL agent. In the second scenario, the model management support is more extensive. Given the model that has been chosen for execution, data required to instantiate the model is elicited from the user. This data is formatted in the appropriate syntax and combined with the model schema made available by the model provider. The instantiated model is then executed at a modeling environment made available by yet another provider and the results returned to the consumer. These tasks are performed by the GAMS agent. Clearly, the tasks performed by the GAMS agent are knowledge intensive—they demand detailed metainformation about the models listed in

which the model is to be executed. Meta information is also required by the AMPL agent in order to invoke the AMPL environment. But, since more knowledge is assumed of the consumer, less metainformation about the AMPL environment is sufficient. However, in both cases this raises the questions of where this meta information is to be obtained, from whom, and how? As we shall see, the answers to these questions have implications for the barriers to entry for consumers and providers in DecisionNet. Before we discuss the detailed answers to these questions in the context of the AMPL and GAMS agents, we present an overview of the technical architecture of DecisionNet to provide the requisite framework.

3 Architecture

Both the scenarios in §2 discussed the execution of decision technologies by consumers after locating these resources in DecisionNet yellow pages. But how did they get to be listed in the yellow pages?

A critical aspect of DecisionNet is the method for adding technologies to the library. DecisionNet is an "open" system in that it allows, in principle, any provider of a decision technology to enter information about their technology into the collection. The only condition is that the technology be executable over the Web. The process of doing so depends on whether the object is to be registered as an "independent" technology or as an "exclusive" technology. An independent technology in DecisionNet is one for which its provider has completely crafted the software required to interface the technology for remote execution; the technology can then run independently of DecisionNet, but consumers and providers can still avail of DecisionNet's yellow pages functionality if it is registered with DecisionNet. For such technologies, providers need only register their technology with DecisionNet, giving metainformation required to invoke the technology by the yellow pages (e.g., natural language descriptions).

An exclusive technology is one which can be used only via the DecisionNet environment. For such technologies, a DecisionNet registration agent leads the provider through a series of steps resulting in both a listing on the DecisionNet yellow pages and in the automated creation of a Web-based user interface to the technology. After registering the appropriate protocol to invoke their technology (e.g., anonymous telnet or the POST method of the HTTP protocol) and declaring certain metainformation (e.g., list and type of input fields) about the technology, providers maintain their own servers, leveraging the distributed nature of the Web and permitting scalability. At run-time, DecisionNet agents use the metainformation obtained at registration time to create an interaction with the consumer, launch execution of the technology, and return the results to the consumer.

Architecture

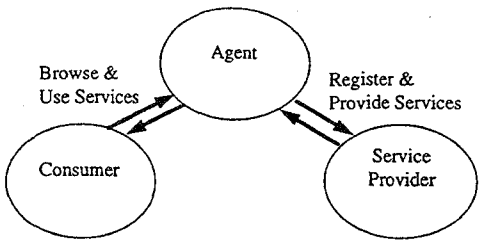

Figure 1: The DecisionNet Architecture

Irrespective of the type of technology (exclusive or independent), the DecisionNet architecture consists of three types of players—consumers, providers and software agents (see Figure 1). Providers register their decision technologies with the software agents that maintain the yellow pages in DecisionNet. During the registration process, metainformation is supplied by the decision technology provider to the agent. This metainformation is minimal in the case of independent technologies, and quite extensive in the case of exclusive technologies. In turn, the agents provide minimal support to users and providers of independent technologies while providing considerable support for execution of exclusive technologies. In the following sections, we discuss the design and implementation of two agents in DecisionNet—the *AMPL agent* which is designed to assist registration and use of an *independent technology—AMPL* and the *GAMS agent* which is designed to assist registration and use of an *exclusive technology—GAMS*.

4 The AMPL Agent

AMPL is used to refer to both a mathematical programming modeling language as well as a computational modeling environment [8]. Models are formulated and stated in the AMPL modeling language.

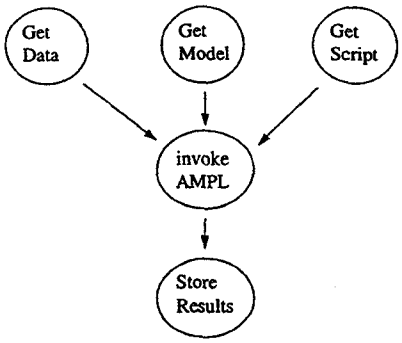

Figure 2: Computational plan of an AMPL agent

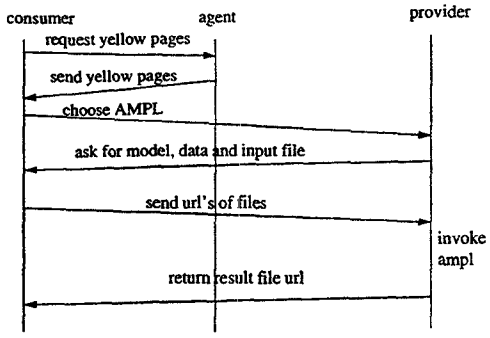

Figure 3: Time sequence diagram of interaction with an AMPL agent

The AMPL modeling environment is a computational platform for instantiating these models with data sets, for selecting and executing solver software, and for formatting and displaying results. The typical session with AMPL on a user desktop consists of the following steps.

Create a file—the model file—containing the model schema that contains the objectives, constraints, and variables associated with the problem to be solved.

Create a file—the data file—containing the data that instantiates elements of the model schema.

Create a file—the command file—containing commands to select, configure and execute a solution algorithm.

Invoke the AMPL environment from the command line with the model file, data file and command file as parameters.

Analyze results returned in a result file, make necessary modifications to the model, data or command files and re-invoke the environment.

How might one make this technology available to a user in DecisionNet? In contrast to the desktop use scenario where all the resources are available on a single platform, in the DecisionNet setting the models, the data, and the modeling environment itself may be on different nodes on the network. Recall that AMPL is an example of an independent technology with the set of assumptions about the consumer discussed in Section 2.1.

4.1 AMPL Agent Design

Given these assumptions about the consumer, the provider independently implements an agent which executes the computational plan shown in Figure 2. The agent performs the following functions, each of which requires it to have certain metainformation. This information may be procedural knowledge and/or instantiating data for these procedures.

1. The agent should be able to retrieve the model, data, and command files needed to run an AMPL session using anonymous FTP or HTTP. In order to do this, the agent must know a) that these three types of files are needed, b) the location (URL) for these files, and c) the internet protocols needed to transfer these files to the AMPL server.
2. These files need to be assembled and used as arguments to a command to invoke the environment.
3. The agent should be able to store the result file and return its URL to the user.

To translate these functional requirements into detailed design specifications, we developed time sequence diagrams which depict interaction between the consumer, agent and provider (see Figure 3). Typically, these diagrams are used with an object-oriented analysis and design methodology [12] and make explicit the methods that need to be invoked to make the interaction possible.

These time sequence diagrams define the following pattern of consumer-agent interaction.
1. Upon invocation from the yellow pages, the AMPL agent sends a consumer an HTML form requesting the

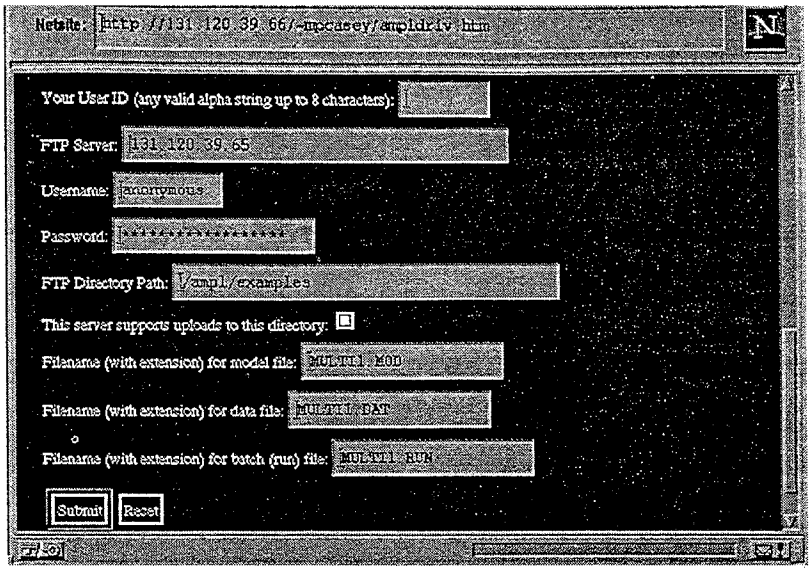

Figure 4: User Interface to the AMPL agent. The user needs to supply the filenames and URL's of data, model and command files.

URL's of the AMPL model, data and command files (see Figure 4).

2. The consumer supplies this information and submits the HTML form.

3. The AMPL agent uses its knowledge of internet protocols to retrieve each of the specified files and invokes the AMPL environment from the command line using these files. The result file is stored on the AMPL server.

4. The AMPL agent returns an HTML page containing the URL of the result file which may be retrieved by the consumer.

This design was implemented as a CGI application using the Delphi environment.

4.2 Implications of the AMPL Agent Design

4.2.1 Registering the AMPL Modeling Environment

Since the AMPL agent is bundled with the AMPL modeling environment, registering the AMPL agent with the DecisionNet yellow pages effectively registers the AMPL modeling environment. This registration process records metainformation about the AMPL modeling environment (e.g., that it is an environment for mathematical programming, the names of its owner, when it was added to the DecisionNet library and so on) and the HTTP call required to invoke it.

4.2.2 Executing the Technology

The consumer locates the record containing information about the AMPL modeling environment in the DecisionNet yellow pages. When the consumer chooses to execute the environment from the yellow pages, the AMPL agent is invoked. From here on, as seen in Figure 3, all interactions are directly between the consumer and the AMPL agent and do not involve DecisionNet.

4.2.3 Implications for Consumers

Consumers are assumed to know both mathematical programming and AMPL. They are required to supply the necessary model, data and command files in AMPL syntax (see Figure 4). Thus, the AMPL agent is designed to support a high end knowledgeable user.

4.2.4 Implications for Providers

The development of the AMPL agent required its provider to combine technology specific knowledge with knowledge about internet protocols for remote execution and file transfer. In terms of our design objec-

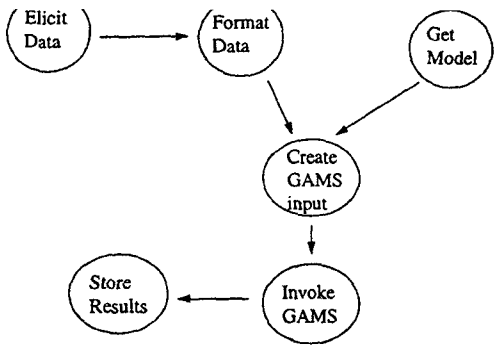

Figure 5: Computational plan executed by the GAMS Agent

tives, the drawback is that this raises the barriers to entry into DecisionNet for providers. However, the advantage is that independent technology providers can quickly get on board DecisionNet in case the market does not provide infrastructural services tailored to the technology.

4.2.5 Implications for DecisionNet

The design of the AMPL agent implies that advertising on the DecisionNet yellow pages is the only service provided by the market. Once the consumer chooses the AMPL modeling environment from the yellow pages, the interactions that follow are directly between the AMPL agent and the consumer. This limits the extent to which market-level infrastructural services, such as billing and state maintenance, can be made available by DecisionNet.

5 The GAMS Agent

Like AMPL, GAMS (Generalized Algebraic Modeling System) [6] refers to both the modeling language and modeling environment for algebraic modeling and mathematical programming. Models are stated in the GAMS modeling language and solved using solvers integrated into the GAMS modeling environment. The typical session with GAMS on a user desktop consists of the following steps.

Create a file—the model file—containing the model schema that embodies the objectives, constraints, and variables associated with the problem to be solved. In addition to the model, this file is also required to order of these elements is important. The first section contains the data, the second section contains the model schema and the file section has the commands which direct the modeling environment.

Invoke the GAMS environment from the command line with the model file as a parameter.

Analyze results returned in a result file, make necessary modifications to the model, data or command section in the model file and re-invoke the environment.

As with AMPL, we consider the design of a GAMS agent that would allow users to interact with the GAMS environment. Due to the fundamental similarities between AMPL and GAMS, this agent could be designed in a similar way as the AMPL agent. However, in this case we explain a different approach—one that results in more functionality but involves a lot more complexity.

To motivate our approach, we note that while a GAMS agent developed in the same way as the AMPL agent could be designed by a provider of the GAMS modeling environment, it would not support one to independently be a provider of a GAMS model. Other providers of GAMS models would need to re-implement such agents in order to provide access to their models. This not only imposes a burden on model providers, but also fails to exploit similarities between all models written in the same language. Under the approach we explain in this section, the GAMS agent automatically produces the same functionality, requiring model providers only to supply certain declarative metainformation about their model schemas. We explain this design in this section.

5.1 GAMS Agent Design

Given these assumptions about the consumer, DecisionNet implements an agent to execute the computational plan in Figure 5. The agent performs the following functions, each of which requires it to have certain metainformation. This information may be procedural knowledge and/or instantiating data for these procedures.

1. Given the identity of a model chosen by the consumer, the GAMS agent should be able to elicit data required to instantiate the model from the user. To do this, the agent must know the sets, the parameters and the tables that need to be instantiated. Further, it needs to know that tables and parameters are indexed

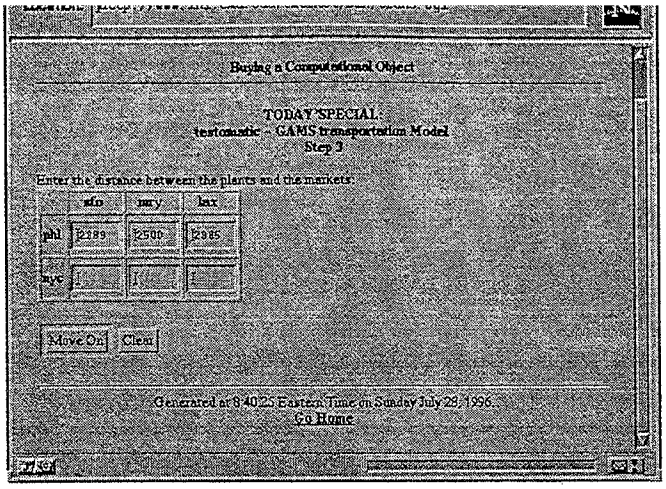

Figure 6: Setting up an instance of the transportation model. The agent presents this form to the user using information obtained on a previous form.

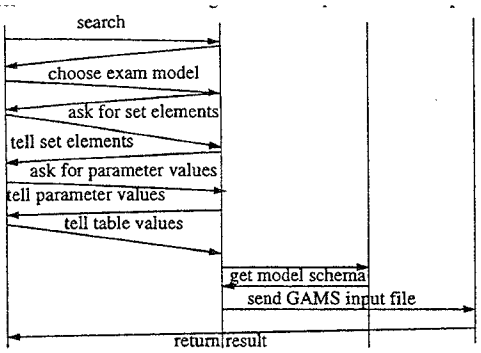

Figure 8: Time sequence diagram of interaction with the GAMS agent

by the sets. This latter information allows it to customize an interface for data elicitation. For example, Figure 6 displays a custom 2×3 table interface created by the agent using the user-supplied dimensions and elements of the sets that index the table.

2. The data elicited from the user has to be formatted in GAMS syntax. Note, unlike the AMPL environment, the interaction with GAMS requires a single file which contains the model, the data and the commands. To create this file, the elicited data needs to be concatenated in the right order with the pre-existing model file made available by the provider. For this the agent needs to know the syntax of the GAMS data section and that it precedes the model section in a GAMS input file. Figure 7 shows a fragment of the data section generated by the agent for a particular model selected during a consumer session.

3. The agent should be able to retrieve the model file chosen by the consumer in order to create the input file for the GAMS session using anonymous FTP or HTTP. The agent must know the location (URL) for the model file, and the internet protocols needed to transfer it to the DecisionNet server.

4. The agent should be able to send the input file to the GAMS server and launch execution from the command line. The agent needs to know a) the internet protocols for file transfer and remote invocation, and b) the invocation command for GAMS.

5. The agent should be able to return the URL of the result file to the user.

These requirements, in turn, raise requirements for information that needs to be elicited from the provider of the model schema and the modeling environment at registration time. In particular, the model schema provider needs to supply metainformation about the model that can be used by the agent to meet the first requirement of eliciting data required to instantiate the model from the consumer. The modeling environment provider needs to supply the metainformation about the GAMS environment in the form of GAMS syntax required by the GAMS agent to format the data elicited from the consumer. The GAMS agent should also be capable of fetching the model file using an internet protocol such as FTP and creating the complete model input file by concatenating it with the data elicited from the user. This knowledge of internet protocols is also required to launch the GAMS modeling environment (which is on a remote server) with the appropriate parameters.

To translate these functional requirements into detailed design specifications, as with the AMPL agent, we developed time sequence diagrams which depict interaction between the consumer, agent and provider (see Figure 8).

The time sequence diagram translates into the following pattern of interaction between the consumer, the agents and the providers.

1. Upon invocation of the model schema from the yellow pages, the GAMS agent uses metainformation to begin data elicitation. Meta information associated with the modeling environment is used to determine

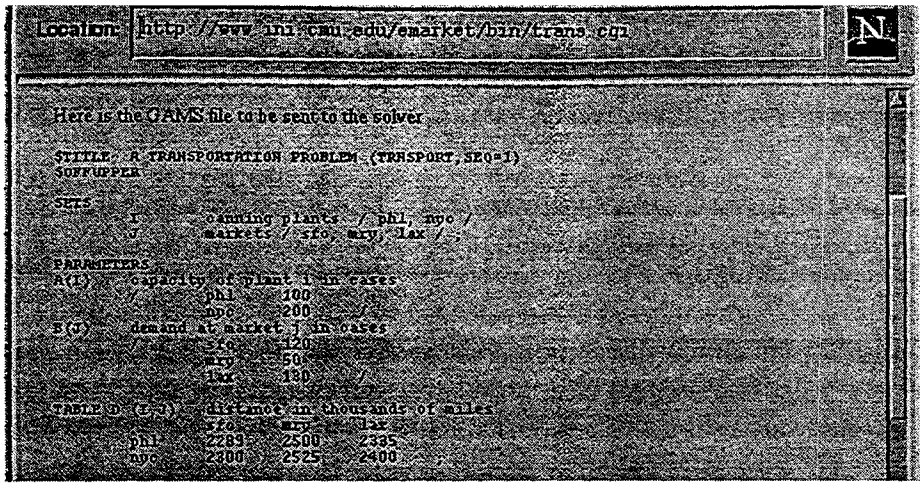

Figure 7: GAMS model and data file generated by the agent using data obtained from the user.

the order in which data is elicited (i.e., information about sets is requested first, followed by information about tables and parameters). This data elicitation uses HTML forms. The forms are dynamically generated with the layout of the form using meta information associated with the model schema. For instance, if there are two sets required to instantiate the model schema, the form that elicits the elements of these sets will contain a section for each set. Forms required to fill out the tables will use the size of the sets declared by the consumer to provide a customized interface to the consumer (see Figure 6).

2. After the data has been supplied by the consumer, the GAMS agent uses information about GAMS syntax to format the data in the GAMS format.

3. The GAMS agent uses its internet agent to retrieve the model schema file from the FTP server maintained by the model provider.

4. The GAMS agent concatenates the data section that it generated based on consumer interaction with the model schema file to create the model input file required by the GAMS modeling environment.

5. The GAMS agent then invokes the GAMS environment with the model input file as parameter. This invocation sequence is based on metainformation supplied by the GAMS modeling environment provider.

6. The result of the GAMS modeling environment execution is returned by the environment to the GAMS agent which redirects this stream to the consumer's Web browser. Alternatively, the result file could be stored on the HTTP server and a URL of the file returned to the consumer.

This detailed design was implemented as a CGI application using the Perl programming language.

5.2 Implications of the GAMS Agent Design

5.2.1 Registering the GAMS Modeling Environment

Since the GAMS agent is required to generate the data elicited from the consumer in GAMS format, considerably more metainformation needs to be supplied by the provider to the agent than in the case of AMPL registration. In addition to descriptive information and the commands required to invoke the environment, information about the syntax of the data section of GAMS needs to be supplied by the provider. Furthermore, semantic interdependencies which determine the *order* in which data needs to be elicited also need to be supplied by the modeling environment provider (e.g., since tables and parameters in GAMS are indexed by sets, the agent needs to obtain information about sets first, then tables and so on). Clearly, this imposes additional demands on the modeling environment provider. However, it enables a user who does not have knowledge of GAMS to use the environment.

user was to execute her models and data on the environment. In the case of GAMS, the objective of the consumer is to select and use a *model schema* such as the examination scheduling model while the objective of a GAMS model provider is to register such schemas. Meta information about the model schema is required by the agent to tailor a data elicitation session specific to the model schema chosen by the consumer. This meta information, in addition to the descriptive narrative of its capabilities, describes the number of sets, tables and parameters that need to be instantiated to execute the model. The consumer will be prompted by the agent to specify the elements of the sets, the cells of the tables using an interface that make GAMS transparent to the consumer. Note that this sort of metainformation was not required in the case of the AMPL interaction. Once again, while the model schema provider is required to supply additional metainformation than in the case of AMPL, the GAMS agent is able to allow users who have no knowledge of GAMS to supply the data and execute a model.

5.2.3 Executing the Technology

The consumer locates the record containing information about the GAMS model schema and requests execution of the model. This request invokes the GAMS agent which mediates all interaction between the consumer and the providers (i.e., the model schema provider and the modeling environment provider). Meta information associated with the model schema and the modeling environment are used in the data elicitation process as well as to format the data in GAMS format.

5.2.4 Implications for Consumers

Consumers are not assumed to know mathematical programming or GAMS. Their objective is to select and execute models listed on the DecisionNet yellow pages. They are only assumed to possess a Web browser capable of working with forms and to be in position to supply data when prompted. This lowers the barriers to entry to consumers.

5.2.5 Implications for Providers

The GAMS agent is provided by the DecisionNet market. While this reduces the work that needs to be done by providers to make their technologies available on the Web, it imposes additional work during

and modeling environment providers to HTTP servers or anonymous telnet servers. Overall, the registration service provided by the GAMS agent reduces the barriers to entry for providers into DecisionNet.

5.2.6 Implications for DecisionNet

The GAMS agent provides a value added service in addition to advertising on the DecisionNet yellow pages. In contrast to the case of AMPL where consumer provider interactions do not involve DecisionNet, the GAMS agent mediates all interactions. This has the advantage of providing additional centralized services such as billing and state maintenance.

6 Discussion

We have described the designs of two different agents that facilitate interaction with optimization models and environments. We have seen that the GAMS agent requires much more metainformation than does the AMPL agent. Does this translate into better functionality for DecisionNet consumers and providers?

With both GAMS and AMPL, there is a hierarchical relationship between the language, model schemas written in the language, and the model instances (or problems) generated by populating a model schema. The AMPL agent, as described, has no knowledge about this relationship. It simply provides a convenient and easy interface to a remote AMPL modeling environment for consumers who are knowledgeable in the use of AMPL. The agent is relatively simple to implement and provides a scalable way in which a technology provider can get listed on DecisionNet.

The design of the GAMS agent, on the other hand, exploits the relationship between the GAMS modeling language, model schemas, and specific model instances for these schemas. That is, the GAMS agent has explicit knowledge about a) the syntax of the GAMS language (e.g., that sets are declared as `set` *set name* `/element1, element2,../;`), b) the relationship between index sets and other structures such as tables (e.g, the dimension of a table is determined by the dimension of its index set) and c) the procedure required to instantiate a model schema (e.g., instantiate sets before instantiating tables).

Therefore, our GAMS agent can actually be viewed as operating at two levels—the GAMS modeling language and GAMS model schemas. In each case the

creation and execution of the model instance. The agent supports the user in doing so by supplying customized input forms for the model schema. But, more interestingly, execution of the GAMS language object results in creation and registration of a model schema which can then be made available via DecisionNet to consumers. The run-time behavior of the agent for instantiation and execution of a model schema is completely *derived* by combining the agent's explicit knowledge about GAMS (and internet protocols) with the model-specific information obtained at model registration (that is, language execution).

What this means is that the GAMS agent can be used by DecisionNet providers of GAMS models to add these model schemas to the DecisionNet yellow pages, with no need for further programming or the development of an interface for the model. This agent functionality has a price in terms of the complexity of its implementation. However, the lessons learned in developing the GAMS and AMPL agents should allow us to develop similar agents for other modeling environments. Going beyond that, it would be interesting to examine whether it is possible to generalize some or most of the knowledge and reasoning performed by such agents. Our current research attempts to develop DecisionNet agents that in turn would be used by modeling environment providers to automatically generate specific agents specialized to the needs of these environments.

References

[1] Sulin Ba, R. Kalakota, and A. B. Whinston. Executable documents as the basis for DSS. In Tung X. Bui, editor, *Proceedings of the Third ISDSS Conference, Vol. II*, pages 373–381, Hong Kong, 1995. International Society for DSS.

[2] P. Becker. A framework for providing and using algorithms and algorithmic meta knowledge on the internet. In Sudha Ram and M. Jarke, editors, *Proceedings of the Fifth WITS, Amsterdam, Holland.* RWTH Aachen, Fachgruppe Informatik, 1995.

[3] Tim Berners-Lee, R. Cailliau, A. Luotonen, H.F. Nielsen, and A. Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, 1994.

[4] H.K. Bhargava, R. Krishnan, and R. Müller. On parameterized transaction models for agents in

[5] H.K. Bhargava, R. Krishnan, and R. Müller. Decision support on demand: Emerging electronic markets for decision technologies. *Decision Support Systems, forthcoming*, 1996.

[6] Johannes Bisschop and Alexander Meeraus. On the development of a general algebraic modeling system in a strategic planning environment. *Mathematical Programming Society*, pages 1–29, 1982.

[7] Anthony Brooke, David Kendrick, and Aleaxander Meeraus. *GAMS: A User's Guide, Release 2.25.* The Scientific Press, 1992.

[8] Robert Fourer, David M. Gay, and Brian W. Kernighan. A modeling language for mathematical programming. *Management Science*, 36(5):519–554, 1990.

[9] M. Goul, A. Philippakis, M. Kiang, D. Fernandes, and B. Otondo. Towards a client/server open–dss protocol suite for automating DSS deployment on the world wide web. In Tung X. Bui, editor, *Proceedings of the Third ISDSS Conference, Vol. II*, pages 517–528, Hong Kong, 1995. International Society for DSS.

[10] M. Jeusfeld and Tung Bui. Interoperable decision support system components on the internet. In Sudha Ram and M. Jarke, editors, *Proceedings of the Fifth WITS Amsterdam, Holland*, pages 56–67. RWTH Aachen, Fachgruppe Informatik, 1995.

[11] R. Krishnan, R. Müller, and P. Schmidt. Accessing "computable" information over WWW: the MMM project. In Tung X. Bui, editor, *Proceedings of the Third ISDSS Conference*, Hong Kong, 1995. International Society for DSS.

[12] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Loresen. *Object-oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, NJ, 1991.